# Accessible GIS Visualizations

Kaitlin Duck Sherwood*

University of British Columbia

**Abstract**

There has recently been an explosion of tools that allow people to interact with map-based data on the World Wide Web, using Google Maps' API. Aside from sattelite and ariel photography, these applications have been *point-based* or *line-based*, not *area-based*. This paper describes a prototype of a system for visualizing area-based data, combining U.S. Census Bureau population density information with Google Maps.

**CR Categories:** H.2.8 [Database Management]: Database Applications—Spatial databases and GIS;

## 1 Introduction

There is an abundance of United States GIS data that is freely available. While The Crown retains rights to works created by Canadian government employees[24], all information collected by employees of the United States government is, by statute, in the public domain[9].

Visualizing the data, however, has traditionally not been practical for people who are not GIS experts. To visualize the data has meant either

1. buying a commercial product like ArcInfo[8] by ESRI (which is sophisticated enough that it is not possible to purchase and download directly),
2. finding and installing an open-source GIS package and understanding the data sets like inovaGIS[18], or
3. writing your own GIS visualizer.

The Web has reached a maturity level such that it is possible to deliver GIS data to lay users. However, as I will discuss in the next section, this work is almost exclusively restricted to displaying point sources of information.

This paper will discuss an application that makes area-based data available. (By area-based, I mean data where there are a set of areas, each with a unique associated data value.) This application does so by rendering polygons with colors based on the polygon's unique data value, then combining that with Google Maps. While this paper's particular application uses U.S. Census Bureau information, any kind of area-based data could be displayed in this manner: school district average test scores, county sales taxes, water district-based per capita water usage, etc.

## 2 Related work

With the Web, people have been starting to make GIS data available to the masses as Web services. MapQuest[23], Yahoo Maps[17], Google Maps[14], Microsoft's TierraServer[16], and Google Earth[13] have all made maps available to non-experts.

Google has an API to its services that has become quite popular. (Microsoft and Yahoo have APIs to interface with their services as well, but they have not spawned as many derivative works.) There are even at least two sites devoted to reporting Google mashups[26][1].

However, these mashups almost all display collections of points, with a virtual pushpin stuck into the map at points of interest, including locations of craigslist housing listings[28], UFO sightings[27], crime incidents[12], and traffic accidents and road construction[25].

There are a few Google mashups that display paths through the map, including traceroute[31] paths and an exercise distance calculator[22].

I found one set of web applications that outlines an area, e.g. a ZIP code boundary[7].

I was only able to find one mashup that reported data collected over an area, but it uses the map only to let the users indicate which point they were interested in, then displays the data separately as text[2].

There is a site that displays NY subway information as an overlay[20].

There is one site that displays two Google maps simultaneously[11], with the technique documented elsewhere[19].

As far as I can tell, there is no mashup that displays area-based information on interactive Google maps.

## 3 Description of solution

When using this as-yet-unnamed web service[29], the user sees two squares, as shown in Figure 1.

The center square is a translucent map showing census tract polygons. This center area acts sort of as a magic lens to show population density. Each polygon in the inner square is filled with a color determined by the population density, with a higher saturation for higher density.

Buttons along the upper right hand side of the larger map control what is rendered on the larger map. The standard Map, Satellite, and Hybrid are there, as well as a new "Census" button. When the user presses the Census button, the back map changes to show population density, just as in the center square.

Each square's map can be dragged independently. At the end of the drag, the window that was not dragged is recentered to align with the dragged window. While this means that the two maps behave in somewhat opposite directions – dragging the center map to the right means that the outside map will move to the left – this is a limitation accepted in order to offload a significant amount of work to Google, which will be discussed later.

*email: ducky@webfoot.com

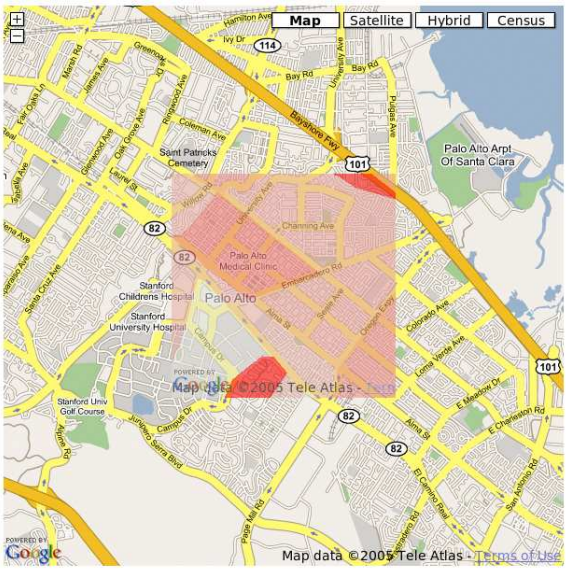


Figure 1: Screenshot


Figure 2: San Quentin

## 4 High-level implementation

### 4.1 Datasets

I got my data from the U.S. Census Bureau. As mentioned earlier, Canadian census information is owned by The Crown. UBC students can use it for academic purposes, but "the distribution of any data obtained under this agreement outside this educational institution through sale, donation, transfer or exchange of any portion of these data in any way is strictly prohibited."[30]

It turned out to be very useful to use a data set for a region that I was personally familiar with. Not only was I able to do a visual sanity check, but I was able to understand some anomolies better. For example, when looking at the population density distribution, I noted a few densities that seemed outrageously high. When I examined California Census tract 122 000 on the map, by being familiar with the area, I was able to realize that I was looking at San Quentin state prison, as is visible in Figure 2.

I was also able to notice some anomolous areas. In addition to two tracts near Palm Springs that listed thousands of people living literally on the freeway in the desert (see Figure 3), the barely-inhabited Farallon Islands are listed as having a population of 1 518 people, and 4 250 allegedly live

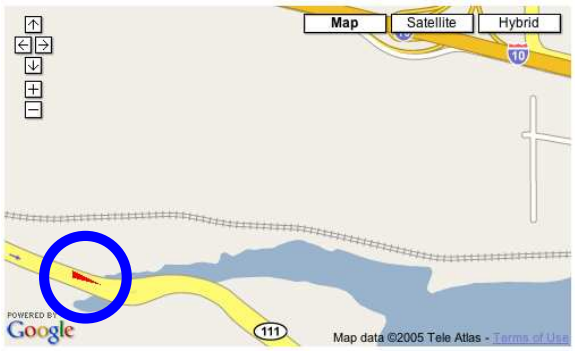


Figure 3: Census tract on a freeway

at the mouth of a channel in Foster City.

To do this kind of mapping, I needed two sets of data from the Census Bureau: information about the shapes and locations of census tracts and the population in each tract. The Census Bureau gives that information in many different forms. I chose to use the California Tract Boundary shapefile[5] and the SF1 population information file[6].

The boundary file had extra columns with the area and the perimiter of the tract. While the file README warned that those columns were generated as a side effect of an internal project and we not to be used, I checked several tracts against the area column and found that they were the area in degrees. Note that while one degree N-S is a constant length, one degree E-W varies with the latitude. With a simple formula, I could transform the degree-squared area into miles-squared areas.

(Note that the anomolies listed above were *not* related to inaccuracies in the area column. The population of the Farallon Islands and the Foster City slough were from the population file, and the *shape* of the Palm Springs freeway tract was wrong.)

### 4.2 Code

This unnamed application has a server-side piece (which generates the map of population density) and a client-side piece (which combines the population map with Google Maps).

I wrote the server-side code in C++, using several public-domain libraries.

1. I used the Shapefile C library[32] to parse the Census Bureau shapefiles and population data files.

2. I used the gd library[3] to draw the polygons (as given in the shapefiles), filled with a color chosen based on the population (as given population data files) divided by the area (as given in the shape file). I wrote files out as PNG images.

3. I used the cgihtml library[21] to connect the program to a Web server.

I wrote the client-side code in JavaScript, using the Google API to combine the population density map with the Google service, extending a technique developed by Adrian Holovaty[11] and documented by Will James[19].

### 4.3 Pre-fetching

The separate dragging of the two different windows is suboptimal, but I had reasons for the design decision that led

to that.

For each zoom level, Google divides the world into 256 pixel by 256 pixel tiles in a Mercator projection. When a user displays a map area, Google fetches the tiles in the user's view *and* pre-fetches the neighboring tiles. It caches those tiles so that they can be presented to the user quickly when he or she moves the map to expose an adjoining view. Those cache requests originate in the JavaScript on the local machine. Without reverse-engineering some code that Google takes some pains to keep hidden, replicating that functionality on the client is a significant job.

In order to take advantage of this caching and pre-fetching, the image being cached and pre-fetched must be a map, not an overlay. Maps are background, overlays are foreground. Maps do not, in the Google model, have things behind them.

Adrian Holovaty uses a clever capability of cascading style sheets[11] on his site, where he creates two Google maps and uses the browser to manage the transparencies, via a CSS "style" in the enclosing of the HTML "div" elements.

The disadvantage is that these two maps are then separate entities, not linked. Google does not have a way to dictate that while one map is dragged, the other should follow. (It does have events that recognize drag ends.)

There two other, more subtle advantages for my application.

1. At higher (i.e. more zoomed out) zoom levels, the rendering of the population density map is very slow compared to fetching the Google map/satellite/hybrid tiles. This method gives the users a fast means of changing context without having to wait for the slow focus.

2. While the population density map is translucent, it is not transparent. By not having it extend to the edges of the background map, it is easier to examine in detail underlying features: just move them out from under the center square.

#### 4.3.1 Color scale

I started out using a blue-yellow color scale. For the proposal, I started with a scale from Color Brewer[4], but as I needed the colors to be translucent, those scales ended up being light enough that they were indistinguishable. I thus bumped up the saturation, but the balance still didn't look right.

Instead, I switched to monochromatic red of varying saturation. I had planned on using a eight- or ten-step scale, but started out with 128-steps in order to see what the distribution of data looked like. I found that the range of the population density was much higher than I had imagined it would be. Excluding bugs in the data and state prisons, the population density in California ranges from 0 to 98 500 people per square mile.

With such a broad range, it becomes very difficult to see any variation in either rural or urban areas. There is a big difference between (wealthy) populated Atherton and the Emily Renzel wetlands, but that is hard to see with such a big dynamic range. I found that I wanted all 128 levels – and then some!

In the first California picture, Figure 4, tracts with more than 100 000 people per square mile are solid red, tracts with 0 people per square mile are solid white, and all other densities are interpolated.

To a very good first approximation, California is empty.

Figure 4: All California with solid red for tracts with more than 100K people/sq. mi.

Figure 5: All California with solid red for tracts with more than 20K people/sq. mi.

Figure 6: All California with solid red for tracts with more than 100 people/sq. mi.

In the second California picture, Figure 5, all tracts with more than 20 000 people are solid red.

In the third California picture, Figure 6, all tracts with more than 100 people are solid red, and yet there are still large areas which are basically featureless.

The color mappings that proved so abysmal for the rural areas give a good level of detail in urban areas like San Francisco.

However, the color mapping that was inadequately aggressive to show rural areas is already overly aggressive for urban areas.

The wide dynamic range against aggregating information. It is very common for there to be huge differences in density in neighboring tracts, as in the case of San Quentin. San Quentin is visible in the (non-scaled) non-aggregated all-California picture, yet if I aggregated data, it would get smeared into the lower level of its neighborly tracts.

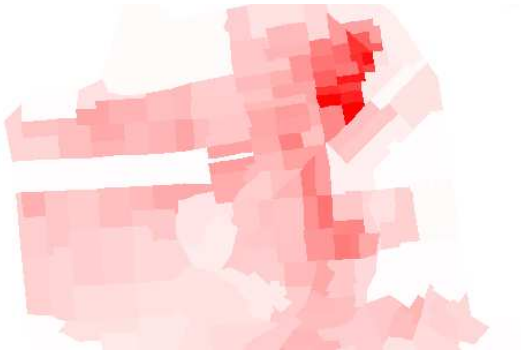

Figure 7: San Francisco with solid red for more tracts with than 100K people/sq. mi.

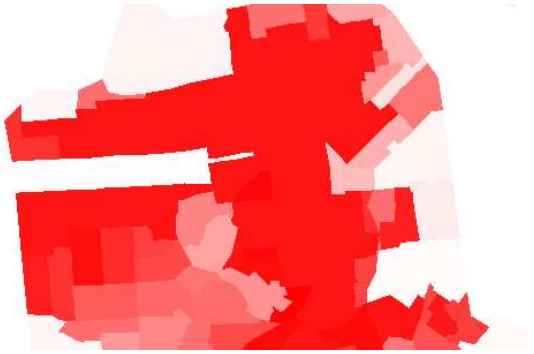

Figure 8: San Francisco with solid red for tracts with more than 20K people/sq. mi.

### 4.4 Caching

I had originally hoped to cache pages in order to bypass the delay in rendering the population density map. For the entire United States, I calculated that to cache all the maps would take on the order of one terabyte. While that is clearly more than a grad student should invest in, it is within the realm of possibility for a small company.

However, if the user needs to select color mappings, then the amount that needs to be cached explodes. If the user can choose between ten minimum cutoff levels and ten maximum cutoff levels, then that's 100 terabytes, and a bit more than companies not named Google would want to spend.

Furthermore, it seems that that amount of disk space would be required for any subsequent service that presented similar area data. School test scores, percentage of non-white residents, and water district per-capita usage would all be another 100 terabytes each.

The good news is that with a small amount of cleverness, this should be a solvable problem.

First, the performance is good enough at low zoom levels that perhaps those don't need to be cached. The rendering time is most strongly dependent on the number of polygons to render. (For obvious proof, try out the application on an area of Eastern California that is very sparsely populated, then look at LA.) As Google doubles the magnification at every zoom level upwards, dropping the bottom two zoom levels reduces the caching needs by 3/4. If zoom level 3 is also dropped, that reduces the space requirements by 7/8.

Second, rendering at the very lowest zoom levels is not very interesting. At zoom levels 1 and 2, even urban census tracts fill the tile completely or nearly completely. Zoom level 3 is only marginally interesting. So even if performance isn't fantastic at lower zoom levels, perhaps nobody will care.

Third, I believe that the expensive operation is the rendering of filled polygons, not in coloring them per se. Imagine rendering one set of tiles, with the color of each tract being set to an encoding of the state and tract index. (California has under 8 000 tracts and has roughly 1/12th of the U.S. population, so the total number of tracts in the U.S. is probably under 96 000, which is well, well within the bounds addressable by a 24-bit color space.) Call those "the geometry tiles".

Now imagine that when the server needs to serve a tile, it reads in the corresponding geometry tile, steps through the color table, extracts the tract index, finds the data for that tract, looks up the color for that data, and inserts that color into the color table at that spot. (This has the effect of changing the color of every pixel without having to do any

calculations.) When the entire color table has been walked, it sends the resulting map to the requesting party.

In addition only having to calculate the geometries of vertices once, this also amortizes the expense of determining which polygons are visible in a tile.

## 5 Results

### 5.1 Color scale

I felt that performance improvements would be more valuable that setting color scales, and concentrated my (futile) efforts there, not on finishing the color scale work.

I have implemented hooks in the server code to allow the setting the "minimum cutoff" and the "maximum cutoff", where any density less than the minimum cutoff would be rendered in white and anything more than the maximum cutoff being rendered in pure red. (In fact, I have had those hooks for a long time, and used them to generate the pictures of California and San Francisco used earlier.)

What remains is to build selectability into the client side code and to build a widget to simultaneously display the current color settings and allow the user to set the ranges.

When Google's code requests a tile, it attaches a query string with the x, y, and zoom values to the base URL that the client-side code specified. However, Google does *not* insert a question mark to separate the query string from the base URL, it requires one be in the base URL. I can thus designate a base URL of the form http://webfoot.com/cgi-bin/foo?minCut=0&maxCut=10000&.

Conceptually, it is very straightforward to add this to the client side code. It is merely a Small Matter of Programming.

Building the widget is slightly more complicated, but not novel and not unique. It appears to also be a Small Matter of Programming.

### 5.2 Caching

I wrote code for changing the color table entry already. It *should* be working already. According to the gd documentation[3], deallocating one color, then allocating another is supposed to put the second color into the first color's newly vacated slot. There is some bug in either my code, the gd code, or the documentation, and I haven't figured out which yet. It seems likely that it is in my code.

### 5.3 Scenarios of use

As described above, the application allows users to pan and zoom through California data with standard Google map controls.

There are a number of possible uses for this application, particularly if one takes a broader view and sees it as a prototype for an entire class of area-based information:

1. Idle curiosity. It is interesting to explore your world, and this is one more way to do so. The census data can point out interesting areas to investigate.

2. Geographically restricting physical-world searches. For example, someone single might choose to look for tracts that had a favorable gender ratio. Parents might look for tracts with high school scores.

3. Data-checking census information. I was quite surprised at first to find errors in the census shapefile. Upon further reflection, however, I realized that without the satellite images backing the census data, I might not have been able to recognize when there was an issue. I found myself frequently wondering about a tract, switching to the satellite view, and seeing an explanation.

### 5.4 Performance

The response time is adequate but not great at lower zoom levels – about half a second system time to render a zoom-level 4 tile of San Francisco (before web transfer delays). At higher levels, it is horrid – on the order of 8 seconds system time and 40 seconds user time to render all of California.

On the other hand, for almost all people, it is much faster than the alternative method of visualizing population densities, which includes earning enough money to buy an ArcInfo license, navigating through open source projects, or writing one's own visualizer.

## 6 Discussion

### 6.1 Lessons learned

Handing in an assignment a week early is very painful.

Working with analog videotape equipment is poor preparation for using Adobe Premiere. Watching other people edit TV shows with Adobe Premiere is also poor preparation for using Adobe Premiere. Furthermore, Adobe Premiere's documentation appears to be written for people who already know how to use it.

### 6.2 Strengths and weaknesses

The application is poor at response times at high zoom levels, as mentioned before.

The application is strong at filling an unexploited niche.

One of the strengths *and* weaknesses of this application is that the work really only needs to be done once. Once there is one application that shows area data appears, others will get the idea. Once one entity produces the geometry tiles, anyone else would be able to use them in their own application

This is great for advancing knowledge and improving the world's access to information, but a lousy value proposition for starting a company.

## 7 Future work

I plan to do the following:

1. Make performance enhancements, as discussed above. This means tracking down the bug in my code and adding a small amount of flow-control code to choose when to write out a tile to disk. Afterwards, I would need to populate the cache of tiles. However, I could choose to do a form of lazy evaluation and not write out a tile to disk until one person asks for it.

2. Write the small amount of code to give the user some control over color mappings.

3. Write the code to dynamically generate a widget that simultaneously displays the color mapping and allows the user to change the color mapping.

4. Make the site more visually appealing and better documented.

If I do the above, and the application proves popular with people not related to me, I would explore expanding it as follows:

1. I am interested in connecting other data sets. Some would take only require changing one line of code to select a different column in the SF1 data file (e.g. Asian population density, Latino population density, density of men, etc.) Another interesting type of extension is to allow interesting ratios, e.g. gender ratio or ratio of single-mother households. To do that stupidly would require about four lines of code change. However, the proper way to do this would be to add a configuration file which listed which files to use and which records in that file to use.

   While XML is the emerging standard for data formats, much of the interesting data is in legacy formats. It might make sense to incorporate the Data Format Description Language[10] into the configuration specification.

2. It would be interesting to explore mapping two variables simultaneously. For example, it might be illustrative to convert median income into a blue color component and ratio of whites to non-whites into a red component. White, red, magenta, and blue would then correspond to the four corners of the income/racial balance matrix.

## 8 Bibliography

### References


[1] aka Cube Monkey. Cool google maps blog. http://coolgooglemaps.blogspot.com/.

[2] LLC AnalyGIS. S.r.c. http://65.39.85.13/google/default.htm.

[3] Thomas Boutell. gdlib. http://www.boutell.com/gd/manual2.0.33.html.

[4] Cindy Brewer. Color brewer. http://www.personal.psu.edu/faculty/c/a/cab38/ColorBrewer/ColorBrewer_intro.html.

[5] U.S. Census Bureau. California census tract boundary data. http://www.census.gov/geo/www/cob/bdy_files.html, 2000.

[6] U.S. Census Bureau. Summary file 1 census bureau population data. http://arcdata.esri.com/data/tiger2000/tiger_statelayer.cfm?sfips=06, 2000.

[7] John Coryat. Zip code boundary map. http://maps.huge.info/zip.htm.

[8] esri. http://www.esri.com/.

[9] United States Government. 17 united states constitution sec. 105.

[10] DFDL Working Group. Data format description language. http://forge.gridforum.org/projects/dfdl-wg.

[11] Adrian Holovaty. Google maps transparencies. http://http://www.kokogiak.com/gmaps-transparencies.html.

[12] Adrian Holovaty and Wilson Miner. chicagocrime.org. http://www.chicagocrime.org/map/.

[13] Google Inc. Google earth. http://earth.google.com/.

[14] Google Inc. Google maps. http://maps.google.com/.

[15] Google Inc. Google maps api. http://www.google.com/apis/maps/documentation/.

[16] Microsoft Inc. Terraserver. http://http://www.terraserver.microsoft.com/.

[17] Yahoo Inc. Yahoo maps. http://maps.yahoo.com/.

[18] inovaGIS.org. http://www.inovagis.org/index.asp.

[19] Will James. Add your own custom map. http://http://mapki.com/index.php?title=Add_Your_Own_Custom_Map.

[20] Will James. Google-ny subway hack. http://www.onnyturf.com/subwaymap.php.

[21] Eugene Eric Kim. Cgihtml library. http://www.eekim.com/software/cgihtml/.

[22] Toby Kinney. Google map tool. http://www.tobyk.com/maps/.

[23] MapQuest. http://www.mapquest.com/.

[24] Government of Canada. Copyright act, revised statutes of canada 1985, c. c-42, s.12.

[25] Greg Woo Ouais. Yahoo-google traffic-weather maps. http://traffic.poly9.com/.

[26] Mike Pegg. Google maps mania. http://googlemapsmania.blogspot.com/.

[27] Poly9. Ufo maps. http://www.ufomaps.com/.

[28] Paul Rademacher. Housing maps. http://www.housingmaps.com/.

[29] Kaitlin Duck Sherwood. Population data google mashup. http://CaCensusOverlay.html.

[30] Data liberation initiative data use license. http://data.library.ubc.ca/datalib/gen/dli.html.

[31] Mapulator. http://www.mapulator.com/.

[32] Frank Warmerdam. Shapefile c library. http://shapelib.maptools.org/.